Computer Science Education
Vol. 17, No. 1, March 2007, pp. 59–83

Routledge
Taylor & Francis Group

An Integrated Framework for Improved Computer Science Education: Strategies, implementations, and results

Leen-Kiat Soh*, Ashok Samal and Gwen Nugent
University of Nebraska-Lincoln, USA

This paper describes the Reinventing Computer Science Curriculum Project at the University of Nebraska-Lincoln. Motivated by rapid and significant changes in the information technology and computing areas, high diversity in student aptitudes, and high dropout rates, the project designed and implemented an integrated instructional/research framework. The framework is based around 10 general design strategies that incorporated administrative, instructional, and research principles. The framework consists of a placement examination, three suites of structured laboratory assignments, multimedia learning objects, and educational evaluation and research designs. The results of implementing the framework in our introductory courses are encouraging and insightful. While validating some of our designs, our research also identified areas for further development and research.

1. Introduction

Continuous and significant changes in computer science (CS) technologies in the areas of software engineering and information technology (IT) create considerable pressure on academic institutions to keep the curriculum current and relevant. Operationally, it requires frequent revisions of course design and adaptations of new textbooks for CS courses. Educationally, it requires measurements and research on how students learn new topics and how these topics impact on the students in both the short and long term. Given the rate of change, it is difficult to implement these changes, and it is particularly challenging to conduct educational research to assess how students are adapting to the new changes. With a systematic method to develop, deliver, and evaluate courses and supplementary activities more students will learn, and learn more effectively. Furthermore, an approach of delivering these courses that is particularly sensitive to varying educational and experiential backgrounds will further promote attracting and retaining students. A CS curriculum that

*Corresponding author. 256 Avery Hall, University of Nebraska, Lincoln, NE 68588-0115, USA. E-mail: lksoh@cse.unl.edu

ISSN 0899-3408 (print)/ISSN 1744-5175 (online)/07/010059-25

DOI: 10.1080/08993400701203782

effectively accommodates both change and growth must be able to overcome intrinsic difficulties associated with rapid change. Concomitantly, it must attract and retain more students into this field despite the increasing technological complexity.

Ever-changing technology also makes it difficult to provide high school students with a consistent and up-to-date coursework foundation. As a result, college level introductory CS courses are filled with students with diverse knowledge and exposure. The need to provide varying levels of remediation is a challenge to both students and educators and is especially challenging with respect to many target populations, including women and minorities (Dabbagh, 1996; Rebelsky, 2000; Sturm & Moroh, 1994). High dropout rates are indicative of the problem: 50% or higher as reported in Allan and Kolesar (1996) and Powers (1999) and 15–30% as reported in Guzdial and Soloway (2002). Difficult to grasp concepts coupled with the skill levels of the students make existing lecture-based education in CS very challenging (Cox & Clark, 1998; Urbain-Lurain & Weinshank, 1999).

In 2002 the University of Nebraska-Lincoln (UNL) Department of Computer Science and Engineering (CSE), in collaboration with the College of Education and Human Sciences (CEHS) and under the auspices of the National Center for Information Technology in Education (NCITE), undertook a major revision of its undergraduate curriculum. Based on a thorough and deliberate process the curriculum was significantly updated to promote rigor and practice and to align with the recommendations in ACM/IEEE Computer Society Computing Curricula 2001. The goal of this process was to prepare students for the dynamic and challenging workplace in the CS and IT fields. The work also included development of an overall framework to conduct systematic research on instructional approaches and their effectiveness in both the short and long term.

The long-term goal of our project was to improve student learning and engagement in CS. First, we envision a CS curriculum that can be intrinsically tested, that is modular, and supported by active learning components such as laboratory assignments and multimedia-based learning objects. Second, we envision that students and instructors can design their own CS and IT courses by sequencing a subset of learning objects and laboratory assignments together, for a variety of educational purposes (e.g. continuing education, distance learning, and post-secondary college preparation). Such a curriculum must have, within each course (a) embedded instructional and educational research designs and (b) effective, flexible, customizable, and modular components.

This paper describes our Reinventing Computer Science Project in terms of the tools, techniques, and instruments developed to improve CS teaching and learning and build towards our long-term goal. We describe how principles from educational psychology and CS were combined to create an improved learning experience for students. While many efforts in the past have developed tools and techniques that relate to one or more aspects of our project, the comprehensive approach to the problem as described here is unique. Teams from education, psychology, and CS working together to improve CS curricula are not common and the collaboration

produces innovative learning environments, assessment at all levels, and mechanisms for revision as necessary.

We first discuss the underlying design strategies that guided the research and development of courseware for our project. Then we outline an integrated instruction/research framework for the introductory course sequence in CS (CS0, CS1, and CS2) that includes a placement examination, structured laboratory assignments, cooperative learning, web-based learning objects, and embedded instructional research. We also present our implementation of the introductory CS courses and discuss the results of the implementation and student assessment. Finally, we offer conclusions and lessons learned that might be generalizable to other curriculum reform efforts that aim to use both technology and educational psychology principles.

2. Design Strategies

Design strategies for our project framework were motivated, in part, by several problems within the CS field. First, rapid and significant changes in software development and IT have a profound impact on CS education, affecting both content and pedagogy. A CS curriculum must have a versatile and up-to-date revision process so that new topics and approaches can be efficiently introduced into the curriculum.

Second, the lack of a standardized K–12 CS curriculum presents significant challenges for high schools to provide students with a consistent and current coursework foundation. From our experience, not only is the level of high school preparation diverse, but also the form and content are inconsistent. High school curricula may cover traditional CS topics such as programming and networking or the focus may be on multimedia and Web page development. As a result, our college level introductory CS courses are filled with students with diverse knowledge and exposure, making it difficult to meet individual learning needs. What is called for is a CS curriculum that can be more effectively customized or tailored to individual needs. There is also the necessity to deliver different levels of introductory CS courses, including remedial courses, as well as the need that college level CS courseware be designed so that it can be delivered to high schools.

Third, in introductory CS courses, especially CS1, dropout rates are generally high, and despite good faith efforts the gender disparity in recruiting, retention, and graduation prevails. Numerous studies have been reported to investigate the reasons for the above, e.g. an inappropriate approach to teaching object-oriented programming in CS1, as reported in Bruce (2004), a lack of engaging projects or assignments, as reported in Guzdial and Soloway (2002), a lack of adequate technology for online learning, as reported in Xenos, Pierrakeas, and Pintelas (2002), and anonymity in a large class with a lack of attention received by the students, as reported in Anderson and Roxa (2000). However, attitudinal variables, such as motivation and self-efficacy, have not been consistently studied, especially in introductory CS courses. UNL's Computer Science and Engineering (CSE) Department, for example, has had senior surveys, exit examinations, and an undergraduate advisory panel to provide insights

into what graduates have learned and experienced. However, like most schools, we do not have a systematic and institutionalized process in place to track the student's progress and provide needed advice and remediation.

With the above motivations, our Reinventing Computer Science Curriculum Project identified the following strategies for revision and implementation of our introductory CS courses.

2.1. Strategy 1

The revision of CS curricula should be phased. Since the introductory course sequence has a significant impact on recruitment and retention, these courses should be revised before other core and upper division courses. This process allows the project to better allocate and manage its resources, evaluate and learn from initial phases, and implement the changes administratively.

2.2. Strategy 2

The project team should include faculty and researchers from computer science and education. To better design the courseware, instructional pedagogy and assessment experts in education are needed to work hand-in-hand with CS faculty.

2.3. Strategy 3

The curriculum should follow authoritative standards, such as the ACM/IEEE Computer Society Computing Curricula 2001. The two leading professional bodies in the field of CS, the Association for Computing Machinery (ACM) and the IEEE Computer Society, have developed guidelines for core topics for a CS degree programme for the past three decades and the guidelines have consistently evolved to better fit technologicaladvances in computing.

2.4. Strategy 4

The curriculum should have modular (and even stand alone) courseware for its courses. This allows curricular modules to be replaced and delivered more conveniently. It also opens up the possibility of customization to fit student needs. For example, if a transfer student has a deficiency in subtopic A he or she could take only the necessary modules on subtopic A, instead of taking an entire course. Some institutions have also developed online modules with the same rationale (Bradley & Boyle, 2004; Herrmann, Popyack, Char, & Zoski, 2004; Herrmann et al., 2003).

2.5. Strategy 5

The curriculum should have flexible and adaptable courseware for students of different aptitudes, motivations, and interests. Modular courseware facilitates

flexibility and adaptability. Further, courseware should be flexible in the way that it is delivered and viewed. For example, online courseware that could be used via the Internet, at anytime and anywhere, is flexible. Modular laboratory assignment activities could also be used individually or sequenced differently.

2.6. Strategy 6

The curriculum should include methods to measure attitudinal variables such as self-efficacy and motivation. Attitudinal measures ascertain student's confidence in their CS knowledge and abilities (self-efficacy) and their intention to continue learning about CS (motivation). Self-efficacy has been shown to be a key student variable, with high correlations with achievement (Schunk, 1989), ease of learning (Schunk & Hanson, 1985), use of active learning strategies (Pintrich & DeGroot, 1990), and instructional persistence (Schunk, 1981).

2.7. Strategy 7

The curriculum should include methods to obtain objective, valid, and reliable student outcome measures (Cross, Hendrix, & Barowski, 2002). The objective assessment allows the curriculum to attain a certain degree of tractability, keeping track of how and what students learn moving from one course to another, especially in introductory CS courses.

2.8. Strategy 8

The curriculum should incorporate hands-on activities, teamwork, collaboration, and cooperation (Gatfield, 1999; Johnson & Johnson 1989; Malinger, 1998; Prey, 1995; Weber-Wullf, 2000), in its introductory courses. Cooperative learning is defined as "working together to accomplish shared goals" (Jensen, Johnson, & Johnson, 2002, p. 161). By doing so, students are not only concerned with their own understanding of the material, but also that of the other group members. The students are working together to achieve the same goal. An advantage of cooperative learning is the development of communication and problem-solving skills (Qin, Johnson, & Johnson, 1995). In pair programming one student is the "driver" (leads the group and is in control of the keyboard) and the other is the "observer" (observes, comments upon, assists, and reviews the work of the driver). This instructional strategy has been shown to have a positive effect on student learning (McDowell, Werner, Bullock, & Fernald, 2003; Williams & Kessler, 2002). Introducing teamwork and hands-on activities into the lower division courses should increase student interest, motivation, and retention and provide preparation for the team-oriented demands of the workforce. When we started the project our CS students in upper division courses had been the only ones who had worked in groups for their team projects.

2.9. Strategy 9

The instructional framework should incorporate educational research. Experiments should be conducted to evaluate different methods of instruction. Research design should be in place to collect empirical data as well as to draw statistically significant conclusions. Formative and summative assessments should also be part of the research design to inform faculty, students, and administrators.

2.10. Strategy 10

The curriculum and its components should be institutionalized, particularly the monitoring and refinement processes, to ensure continuity and quality. Software tools and courseware are kept online, institutional knowledge is updated, instructors are informed of the processes, teaching assistants are briefed to observe and record feedback on the courseware, and the department is committed to implement the designs. It is also important to obtain the support of the various administrative units that oversee the curriculum design and changes.

3. Integrated Framework

Our Reinventing Computer Science Curriculum Project is an integrated framework of courseware and software tools, combined with educational research design. Presently, it includes a placement examination, a suite of structured laboratory assignments, web-based learning objects, evaluation and research designs, and a revision process. The current phase of our project focuses on introductory CS courses, i.e. CS0, CS1, and CS2. Each of these components is described below, along with the embedded research and evaluation designs and results.

3.1. Placement Examination

The primary purpose of the placement examination is to appropriately place students into one of two introductory CS courses offered by our CSE department. The first option is CS0, intended for students who lack prior exposure to logic constructs and fundamental CS terminology and students who are not CS majors but who want to gain a basic understanding of the field of CS, including programming. The second option is CS1, intended for students who have a basic understanding of computing concepts and who are likely to major in CS. Having these two courses allows our CSE department to serve students with diverse backgrounds and different goals.

The contents of the placement examination are guided by the description of the CS1 course in ACM/IEEE Computing Curricula 2001. Table 1 shows the topics covered in the placement examination. The first five groups of topics are used as the placement criteria; the second five are used in the pre- and post-test analysis for CS1. The examination covers each of the major topics recommended by the Computing Curricula 2001 guidelines and tests the students' knowledge of each topic at multiple

Table 1. Selected Curricula 2001 topics covered in the placement examination

G	Topics to be covered	Expected competence
1	Functions, sets, relations [DS1]	Application
	Basic logic [DS2]	Analysis
2	Fundamental data structures [PF3]	Application
3	Fundamental programming constructs [PF1]	Application
4	Algorithms and problem-solving [PF2]	Application
	Fundamental computing algorithms [AL3]	Comprehension
	Recursion [PF4]	Comprehension
5	Machine level representation of data [AR2]	Comprehension
	General knowledge	Comprehension
	Computer Architecture [AR1]	Comprehension
	Programming Languages [PL1]	Knowledge
	Software design [ES1]	Knowledge
	Software tools and environments [SE3]	Knowledge
6	Mastery of fundamental data structures [PF3]	Application
7	Mastery of fundamental programming constructs [PF1]	Application
8	Object-oriented programming [PL6]	Application
9	Fundamental computing algorithms [AL3]	Comprehension
	Recursion [PF4]	Analysis
10	Event-driven programming [PF5]	Application
	Software engineering	Comprehension
	Object-oriented design, reusable classes [SE1]	Comprehension
	Testing fundamentals [SE6]	Comprehension

levels, based on the cognitive model of Bloom's taxonomy (Bloom, 1956). Bloom identified six categories of the cognitive model: knowledge or recall, comprehension, application, analysis, synthesis, and evaluation. Questions on our placement examination cover only the first four categories.

The placement examination has undergone two major revisions and has now stabilized. The revisions were based on several statistics. Item difficulty is determined by the percentage of test takers who answer the question correctly. Questions that are too easy or too difficult do not provide meaningful discrimination. Our target mean for each question was between 0.4 and 0.85. The item total correlation for a question shows the correlation between the students' response to a question and their total score. A good question should have a high positive correlation, i.e. students who have high overall scores should be less likely to answer easy questions incorrectly and students who have low overall scores should be less likely to answer difficult questions correctly. While a value of 0.3 is generally regarded as a good target, we chose a value of 0.2 as acceptable. For multiple choice questions the frequency of response for the choices can also be used to measure the overall quality of a question. For example, choices that are not picked should be redesigned or removed.

We also focused on internal consistency reliability, which is a measure of the item-to-item consistency of a subject's responses within a single test. We used Cronbach's

α to assess the internal consistency reliability of our test. Our examination achieved reliability measures of 0.70 – 0.74, which is considered reliable.

We also computed two predictive validity measures for our examination. First, the overall predictive validity for our examination was determined by correlating a student's total score on the placement examination with his/her examination scores on the course ($r = .77$, $p < .001$). Second, the placement predictive validity is computed by correlating students' scores on the first 25 questions with their total points on the course ($r = .63$, $p < .001$). A good placement examination should have a high predictive validity value.

Presently we are in the process of installing the placement examination to place students in one of three options: CS0, CS1, and CS2. Students who perform poorly in the first five groups of questions are placed in CS0. Students who perform well in the first five groups but not in the second set of five groups are placed in CS1. Finally, students who perform well in all 10 groups of questions are placed in CS2, essentially "testing out" CS1.

Readers are referred to Nugent et al. (2006) for a detailed discussion of the design of our placement examination.

3.1.1. Research design and results for the placement examination. We use the placement examination in a pre-test/post-test research design. At the end of the semester students on CS1 take the examination as part of their final. Thus we are able to measure gains at the individual student level and at the topic level for the class as a whole. We are also able to calculate subscale scores, reported by content area and level of Bloom's taxonomy. The Bloom's taxonomy scores showed that students generally performed best on the comprehension and knowledge questions and lowest on the higher level analysis questions. This result is to be expected, because comprehension knowledge is a precursor to higher level analysis skills. Differences in scores on the comprehension and application pre-test problems across the reported semesters were likely due to the revisions of the examination. For example, in the autumn 2004 semester, application questions involving pseudocode were replaced by problems using actual Java code. In addition, some of the easier comprehension questions from the earlier semesters were made more difficult in an effort to improve discrimination.

On examining content area scores it was observed that students consistently scored highest on object-oriented programming and fundamental programming constructs and lowest on programming constructs involving semantics/syntax and algorithms and problem solving. The low scores on algorithms and problem solving are likely due to the higher order thinking and analysis required for this difficult topic. Pre-test scores on object oriented programming showed an upward trend, suggesting greater coverage of this topic in high school. The high scores on fundamental data structures reflect the lower level knowledge and comprehension skills inherent in this area.

It is also important to note that the scores from pre-test to post-test increased across all levels of Bloom's taxonomy and content area. Results from pre-test to post-test for the CS1 course across two semesters showed a significant increase in

achievement, validating the instructional effectiveness. In autumn 2003 the overall mean for the pre-test was 27.45 (SD = 5.32), which improved to 34.30 for the post-test (SD = 5.34). A *t*-test for the significance of difference between these two mean values was highly significant [$t(63) = 11.04$, $p < .001$, $r^2 = .32$]. Similar results were found for spring 2004 [$t(68) = 11.81$, $p < .001$, $r^2 = .45$]. The overall mean for the pre-test was 27.09 (SD = 5.75), which improved to 33.61 for the post-test (SD = 5.61).

The greatest increases were in content areas emphasized in the CS1 course. In general, we believe the scores provide evidence of measurement validity because they show consistent patterns reflecting what we know about the content knowledge of students entering the programme and what was emphasized in our CS1 course.

3.2. Structured Laboratory Assignmentss

Unlike open laboratory assignments, which tend to be an informal environment provided for students to practice their skills with optional attendance, closed or structured laboratory assignments have mandatory meeting times which support the lecture component of the course. Structured laboratory assignments have several advantages. Students learn at the beginning of their majors to be active learners through goal-oriented problem solving in a laboratory setting (Parker & McGregor, 1995). Laboratory assignments also promote students' cognitive abilities in comprehension and application (Doran & Langan, 1995). Qualitative improvement in student learning in the closed laboratory assignment sections for the CS1 course has also been reported (Kumar, 2003; Thweatt, 1994). Exploration opportunities also help first time programmers overcome common hurdles, such as misconceptions about the nature of computers and programs (Lischner, 2001). The laboratory environment also facilitates cooperative learning among students (Oliver & Dalbey, 1994) and can also help increase student retention (Geitz, 1994).

We have developed a suite of CS0, CS1, and CS2 laboratory assignments. Table 2 shows the central topics covered in the laboratory assignments for these courses.

Each laboratory assignment includes (1) a student handout, (2) a laboratory worksheet, (3) an instructional script, (4) a pre-test, and (5) a post-test.

The student handout serves both as a preparation guide and the laboratory script. It includes the laboratory assignment objectives, a description of the activities that will be performed during the laboratory assignment (including the source code where appropriate), a list of references to supplementary materials that should be studied prior to the laboratory assignment and a list supplementary references that can be reviewed after the student has completed the assignment. The student handout also provides optional activities that can be completed during or following the laboratory assignment to give students an opportunity for extra practice.

During each laboratory assignment students are expected to answer a series of questions for each activity and record their answers on a worksheet (paper). Worksheets contain questions specifically related to the laboratory assignment activities and provide the students with an opportunity to find the answers by

Table 2. Laboratory topics for CS0, CS1 and CS2

CS0	CS1	CS2
Introduction to computer account	Introduction to interactive design environment (IDE)	Introduction to the UNIX operating system and the make utility
Pseudocode and algorithms	Documentation	SQL queries
First C program and dissection	Simple class	Advanced SQL queries
Expressions and I/O	Testing and debugging I	Writing and compiling simple programs (in C++)
Functions and procedures	File I/O	Classes and operator overloading (in C++)
Conditionals	Applets	Pointers and references (in C/C++)
Debugging	Event-driven programming 1	Standard template libraries (in C++)
Loops	Exception	Debugging (in C++, with gdb)
Loops 2	GUI/swing	The Web and scripting languages (HTML and PHP or JSP)
Parameter passing	Event-driven programming 2	Creating web forms (PHP or JSP)
Arrays	Inheritance	Creating an interface to a database (PHP or JSP)
Strings	Testing and debugging 2	Creating an interface to a database (JAVA)
Debugging and testing	Simple UML	Web security
	Recursion	Unified Modeling Language (UML)

programming-based exploration. These worksheets also serve as an assessment tool to gauge the student's comprehension of topics learned and practiced in the laboratory assignment.

In addition to the student handout, the laboratory assignment instructor is provided with an instructional script that provides supplementary material that may not be covered during lectures, special instructions for the laboratory assignment activities, hints, resource links, and notes from previous experience with the assignment. Additional space is provided at the end of the instructions for each activity to allow the instructor to record his or her comments regarding the activity and suggestions for improving the laboratory assignment.

The laboratory assignment pre-tests are administered online and students are required to pass them prior to commencing the assignment, however, students may take each pre-test as many times as necessary to achieve a passing score (80%). The pre-test is in open book and open note format and includes multiple choice, short answer and true/false questions. The goals of the laboratory assignment pre-test are to encourage students to prepare for the laboratory assignment and to allow them to test their understanding of the assignment objectives and concepts prior to commencing it. Questions for the pre-test are taken from a variety of sources, including the course

textbook, other textbooks, and questions found on the web. Questions are categorized according to Bloom's taxonomy (Bloom, 1956).

During the last 10 minutes of each assignment, students take an online post-test as another measure of their comprehension of laboratory assignment topics. Like the pre-test, questions are taken from a variety of sources and are also categorized according to Bloom's taxonomy. It should be noted that the goal of the post-test is to assess how well students learned the concepts after they have performed the activities specifically designed to reinforce the concepts.

The design of our laboratories is modular and flexible and embeds methods for collecting data on student learning. Each laboratory assignment has a set of activities which can be replaced and refined over time. Each laboratory assignment is stand alone and can also be replaced and refined. Indeed, we have added several new laboratory assignments over the past few semesters (Testing and Debugging 2 and the UML laboratory assignments are such examples). Since the laboratory assignment documents are self-contained and do not need additional instruction, each laboratory assignment is flexible and can be delivered in situ in a computer laboratory or at a distance. The pre-test and post-test are also available online.

We are currently in the process of completing the laboratory assignments for the CS0 course. By the end of autumn 2005 we will have a suite of 14 structured laboratory assignments for CS0.

Details of our CS1 laboratory assignments and the embedded instructional research design can be found in Soh, Samal, Person, Nugent, and Lang (2005a).

3.2.1. Research design and results for laboratory assignments. Because there are multiple laboratory assignment sections for CS1 we are able to implement different laboratory assignment designs or delivery methods in different laboratory assignment sections and test to see which are most effective. We have conducted experiments on completion of laboratory assignments by individuals versus cooperative groups.

To examine the impact of cooperative learning we designed three types of laboratory settings to incorporate differing levels of cooperation within the in-class exercises: (a) cooperative structured groups, in which students had formally defined roles; (b) cooperative unstructured groups, in which students did not have defined roles; (c) direct instruction groups, in which students worked individually.

In both cooperative groups students worked in groups of approximately three or four students and the laboratory instructor served as a facilitator to encourage group problem solving, as well as being available to answer questions. The primary difference between the two cooperative groups was whether the structure of the group was formally or informally defined. Students in the cooperative structured group had formally defined roles, which alternated each week. Within this setting the laboratory instructor was responsible for monitoring which student took the role of driver and led the group work and keyed the information into the computer. The other students acted as observers and were expected to be active in problem solving and to share ideas within the group. The goal of the cooperative structured group design was to

develop an interdependence among the group members based on the environment (shared computer) and to break the tasks into smaller parts with each member being responsible for a part. By doing so, the group succeeded if and only if each individual contributed his or her part. There were 55 students in the cooperative group with structure, 65 students in the cooperative group without structure, and 64 students in the direct instruction group.

The cooperative unstructured group was the same as the cooperative structured group except that the students were responsible for assigning roles and completing tasks without intervention by the laboratory instructor. Thus, the roles were assigned informally rather than formally.

The final group served as the control group, in which students worked individually. We randomly assigned the pedagogic treatment (cooperative structured, cooperative unstructured, or direct instruction) to three individual laboratory assignment sections. While this method does not have the advantage of random selection of individual students, equivalence of the sections was verified by examining the mean scores for each section on the course pre-test. Between group tests showed that average score in the placement examination was not significantly different between the three sections [$F(2,63) = 2.54$, $p = .09$].

An initial study was conducted in autumn 2003 to test the impact of cooperative learning in CS1 laboratory assignments. In this study both cooperative learning groups performed significantly better than the direct instruction group [$F(2,66) = 6.33$, $p < .05$], as measured by the final laboratory assignment grade. The mean score difference between the cooperative structured group and cooperative unstructured group was, however, not significant (see Figure 1). In a subsequent study conducted in spring 2004 there was no significant difference between treatment groups [$F(2,64) = 2.41$, $p = .10$]. There was, once again, a trend for the students working in

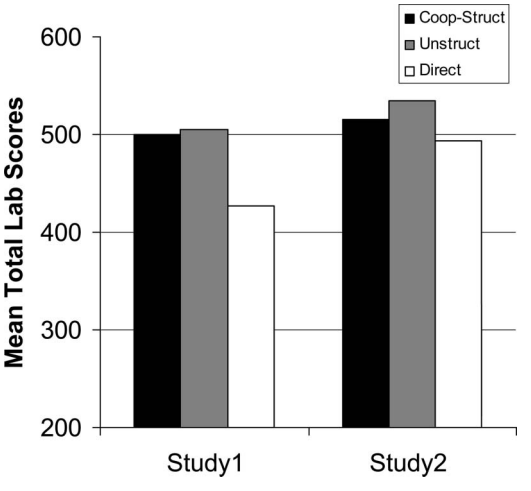

Figure 1. Total laboratory scores (means) for two studies and three types of groups: cooperative group with structure; cooperative group without structure; direct instruction

groups to perform better than the students who worked individually. Thus, we see that cooperative learning is a more effective learning approach than individual learning in our laboratory assignment setting.

Details of this study can be found in Lang et al. (2006).

As part of the ongoing evaluation of our laboratory assignments we also measured students' self-efficacy and motivation before and after the CS1 course (Soh et al., 2005a). This Likert-type survey (Table 3) has statements with five choices: strongly agree (5), agree (4), neutral (3), disagree (2), and strongly disagree (1). Questions 1, 3, 7, and 8 are used to gauge a student's self-efficacy, while questions 2, 4, 5, and 6 are used to measure a student's motivation. These questions were adapted from the Motivated Strategies for Learning Questionnaire (MSLQ), developed by Pintrich and DeGroot (1990). This survey is administered at the beginning of the semester and then again at the end of the semester (pre-test/post-test research design) to measure changes in student self-efficacy and motivation using paired samples *t*-tests. Table 4 shows a consistent decrease in both measures from pre-test to post-test. We also noticed that the self-efficacy measure showed a significant decrease in autumn 2003

Table 3. Survey questions given to the students before and after the CS1 course to measure self-efficacy and motivation

Self-efficacy/motivation survey question
1. I am confident in my CS knowledge and abilities.
2. I am motivated to learn more about CS/technology.
3. I did an excellent job on the problems and tasks assigned for this class.
4. I valued the opportunity to apply what was taught in lecture in a lab setting.
5. I valued the opportunities to interact and collaborate with other students in the class.
6. I was academically prepared to take this course.
7. Compared to other students in the class, I did well.
8. I think I will receive a good grade in this class.

Table 4. Student mean scores on self-efficacy and motivation before and after the semester

	Fall 2003		Spring 2004	
	Before	After	Before	After
Self-efficacy	3.76	3.48[a]	3.63	3.44[a]
Motivation	4.22	3.50[b]	4.19	3.24[b]

[a] $p < .05$.
[b] $p < .001$
Mean scores on the content and Bloom subscales can range from 0 to 1. A mean score of 1.0 would indicate that all students responded correctly to the questions in this content area. The ranges for self-efficacy and motivation scores are 1 – 5, with five indicating high efficacy and motivation.

[$t(71) = 2.08$, $p < .05$], with similar results in the spring [$t(62) = 1.99$, $p = .05$]. The motivation variable exhibited even greater decreases [autumn, 2003, $t(71) = 4.1$ $p < .001$; spring 2004, $t(62) = 5.16$, $p < .001$].

These results clearly show that motivation and self-efficacy decreased in students from the beginning of the semester to the end. This was an unexpected outcome, and we believe it may be due to students' inflated idea of self-efficacy on starting the course. One question from the self-efficacy instrument specifically assessed students' perception of their preparation to take the course. Results from this question indicated that students started the course believing that they were academically prepared, but exited it realizing that they were not.

We were also discouraged that students left the class less motivated to continue in the field of CS and engineering. This result may be confounded by the fact that we did not distinguish CS majors from non-majors. Many non-majors take CS1 to fulfil a science requirement and have no intention of taking more CS classes. This is especially true for the spring semester students. It is possible, therefore, that non-majors were more susceptible to reduced motivation than majors.

We have also developed a second survey to obtain additional feedback from the students about the various components of each laboratory assignment, to determine how, for example, supplementary links provided in each laboratory assignment handout have been used, whether students carried out the pre-laboratory assignment activities, and so on. With this survey we investigated the relationships (a) between what students think of the laboratory assignment content/format and students' motivation to prepare before laboratory assignments and pursue the topic further afterwards, (b) between what students think of how much they have learned and the usefulness of the laboratory assignments, and (c) between the students' motivation and their view of the laboratory assignments, by estimating student motivation based on the amount of time they spent before each assignment and what they thought of the assignment relative to their overall success on the course, and so on.

Based on the surveys, we observed the following. The laboratory design is useful and appropriate in format and content. The laboratory assignment materials are informative and help students learn. Students who were motivated thought that the laboratory assignment materials were useful and were willing to go through the content before each assignment. This indicates that our laboratory assignment design did not have negative effects on student motivation to attend the laboratory assignments (although in another study we found that student motivation to pursue a CS major decreased after taking CS1). However, students who were not motivated to prepare before the laboratory assignment failed to appreciate it. Thus, it seems that the laboratory assignments were unable to increase student motivation for those who did not want to prepare. This insight has prompted us to investigate ways to ''enliven'' some of the assignments. We were encouraged by the observation that student performance in the laboratory assignments correlated with their performance in examinations and homework assignments. This indicates that our design of these components was aligned. The correlation between student performance in the laboratory assignments and on the placement examination is generally not significant.

This matches our expectation, since the placement examination emphasizes a student's comprehension and problem solving, while the laboratory assignments emphasize a student's programming skills. However, we did see that students who performed better in our placement examination are more likely to perform better in laboratory assignments involving more problem solving. These results of the surveys are detailed in Soh, Samal, Person, Nugent, and Lang (2005b).

3.3. Learning Objects

From an instructional standpoint, learning objects are small, stand alone, mediated, content ''chunks'' that can be reused in other instructional contexts, serving as building blocks to develop lessons, modules, or courses. The value of learning objects has been advanced by the Department of Defense (Advanced Distributed Learning, 2003), business and industry (Longmire, 2000), public schools (Nugent, 2005; Pasnik & Nudell, 2003; Pugliese, 2002), and institutes of higher education (Koppi & Lavitt, 2003; Wiley, 2000). Research on learning object approaches has verified its instructional value (Boster, Meyer, Roberto, & Inge, 2002; Bradley & Boyle, 2004). Although learning objects have been in use for more than a decade and much effort has focused on the development of technology and standards, few formal approaches have concentrated on the actual design of learning objects.

Our instructional design approach focused on appropriate use of multimedia elements, student practice, feedback, and guidance, with the goal of encouraging students to be cognitively active while minimizing cognitive load demands (Mayer, 2001). Our learning objects were also designed to be compliant with the Shareable Content Object Reference Model (SCORM; see Advanced Distributed Learning 2004), operated within the University of Nebraska's BlackBoard course management system. Learning objects included the following four basic components, each of which served a specific instructional function: (1) a brief tutorial or explanation provided definitions, rules, and principles; (2) a set of real world examples illustrated key concepts and included worked examples, problems, models, and sample code; (3) a set of practice exercises, including immediate, elaborative feedback, provided important active experiences; (4) a set of problems graded by the computer provided a final assessment.

To date we have built two learning objects: simple class and recursion. Figure 2 shows a screen shot of the module on recursion, which involved a practice exercise. We have delivered the objects as a replacement for two CS1 laboratory assignments and we have required all CS1 students to review these objects to help them prepare for class examinations. Being SCORM-compliant, our learning objects can be deployed within any SCORM-compliant course management system, such as Blackboard. At present we are planning on delivering these learning objects to high school students and collecting data on achievement, motivation, and self-efficacy.

3.3.1. Research design and results for learning objects. To determine the effectiveness of the learning objects we used a control–treatment design. All students in each

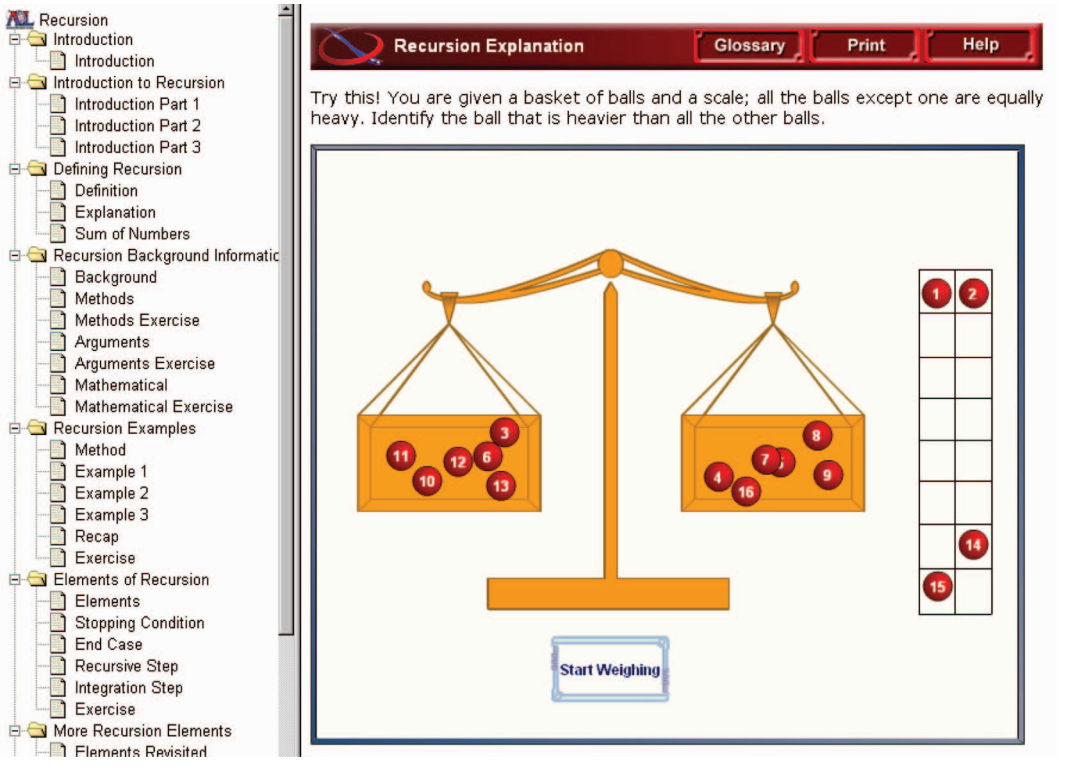

Figure 2. Screen shot of practice exercises component

experimental condition completed the same laboratory post-test, which tested students' understanding of the concept of simple class. Assignment to one of the two treatment groups was made by laboratory assignment section. One laboratory assignment section participated in the traditional laboratory activities; the other spent the laboratory assignment time completing the Web-based learning object. Because individual students were not randomly assigned to the treatment conditions (learning object versus traditional laboratory assignment), equivalence of student CS knowledge and abilities between laboratory assignment sections was especially important. To test for group equivalency we examined the pre-test scores. There was no significant difference between mean placement examination scores for the two laboratory assignment sections [traditional laboratory assignment mean = 26.42, learning object mean = 26.88, $t(48) = .20$, $p = .84$]. Mean score for the learning object group was 7.88 (SD = 1.51); mean score for the traditional laboratory group was 8.29 (SD = 1.23). However, the difference between the two means was not significant [$t(48) = 1.04$, $p = .30$], indicating the approximate equivalence of the learning object to the traditional laboratory experience. These series of evaluation and research results confirm our belief that modular learning objects can be used successfully for independent learning of complex subject matter and are a viable option for distance delivery of course components.

4. Discussion

Here we summarize what we have accomplished over the past 3 years with respect to the design strategies. We discuss the impact of the strategies and whether they were effective or not. Readers interested in an educational and student cognitive perspective are referred to Samal, Nugent, Soh, Lang, and Person (2005).

4.1. Strategy 1. Revision of the CS curriculum should be in phases

We chose to focus on CS1 as our first phase since it is the first core course for our CS and computer engineering majors, as well as being a required course for many non-majors. The course, with a majority of CS majors and with its large numbers, provided us with a large enough sample size to carry out our research studies, which in turn allowed us to refine our instructional designs and materials. The experience we gained from revising our CS1 course provided a valuable insight into the changes needed for CS0 and CS2. We did not need as many revisions of these two courses as was needed for CS1. Following our revision of the introductory courses we have introduced two new upper division courses: CS Senior Design Project and CS Internship. The CS Senior Design Project course, which requires students to work in teams and practice their communication skills, both writing and presentation, has been offered once and will be required for new students. The Internship course will be offered in spring 2006 and will provide students with hands-on experience in a workplace environment and provide opportunities for decision-making, planning, and design of real world solutions.

4.2. Strategy 2. The project team should include faculty and researchers from computer science and education

Our project team consists of students and faculty from the Computer Science and Engineering Department, the College of Education and Human Sciences, and the National Center for Information Technology in Education. This combination of faculty provided a comprehensive perspective regarding content, pedagogical, and assessment issues. To facilitate the collaboration, we started our Reinventing Computer Science Curriculum Project with a semester-long, weekly series of seminars (in spring 2003) involving researchers and students of CS and education. This seminar proved to be beneficial to the participants: both sides learned about each other's terminologies, theories, and strategies, allowing the team to collaborate and understand each other.

4.3. Strategy 3. The curriculum should follow authoritative standards

Our placement examination and the laboratory assignment topics are based primarily on the ACM/IEEE Computer Society Computing Curricula 2001. However, the CS curriculum is constantly evolving. For example, the 2005 version has just recently

been released. Further, due to constraints on departmental resources (faculty, students, etc.), not all topics can be covered to the extent required or recommended by Curricula 2001. As a department we have had to exercise judgement in identifying what topics to cover and at what depth. We also further translated the expected level of understanding of a topic from Curricula 2001 to the equivalent Bloom's taxonomy for assessment.

4.4. Strategy 4. The curriculum should have modular (and even stand alone) courseware

We have produced modular structured laboratory assignments for CS0, CS1, and CS2. We have also built two learning objects. Within each module the design is also modular. For example, individual activities (such as tutorials, examples, and problems) in a laboratory assignment or a learning object can be replaced and resequenced. These modules can also allow other audiences (high school students, community college students, and business and industry employees) to take advantage of course content. Although breaking down a course into modules is not difficult (one could use Computing Curricula 2001 as a guide), initially building a batch of these modules to "fill" a course can be exceedingly resource taxing and time consuming. However, once a base set is developed, adding new, individual modules is significantly easier.

4.5. Strategy 5. The curriculum should have flexible and adaptable courseware for students of different aptitudes, motivations, and interests

We designed learning objects so that students who knew the material could progress quickly while those who needed more practice had the opportunity to obtain additional practice. Although our courseware was built with the goal of flexibility, we did not give students or faculty the opportunity to "customize" courses. The main reason was that we do not have a large enough repository to allow this degree of flexibility. However, we note two different instances that demonstrate the flexibility and adaptation of our courseware. In spring 2004, after noticing that students were not proficient at debugging, even after one Debugging and Testing assignment, we were able to quickly build another Debugging and Testing laboratory assignment for that semester and substitute it into the course. In autumn 2005 several students were unable to attend their laboratory assignment section because of a holiday. We simply required them to take the simple class learning object, which coincided with the laboratory assignment topic for that week. As a result, these students were able to keep pace with the rest of the class.

4.6. Strategy 6. The curriculum should have methods to measure attitudinal variables such as self-efficacy and motivation

We have designed several surveys to assess student self-efficacy and motivation, recognizing that a positive attitude is critical for retention and student success. This is

one area where our research has shown disappointing results; students' motivation and self-efficacy actually decreased from the beginning to the end of the CS1 course. For our future work in this area we plan to distinguish CS majors from non-majors. A CS major will be more likely approach course CS1 with greater seriousness than a non-major, who may be required to take an introductory CS course. We also plan to utilize qualitative research strategies using interviews to obtain more in-depth understandings of students' CS1 experiences.

4.7. Strategy 7. The curriculum should include methods to obtain objective, valid and reliable measures of student outcomes

We have been careful to use instruments with established reliability and validity, and the instruments we have designed ourselves have undergone thorough pilot testing and revision to establish acceptable psychometric properties. We have also used the placement examination, laboratory assignment post-tests, worksheets, and course examations as assessment mechanisms. The placement examination is essentially our workhorse. It not only serves as our placement examination, placing students on either CS0 or CS1, but also serves as the normalization factor for our research studies. One future goal is to develop mechanisms to track students throughout their CS degree sequence, allowing us to better assess the role of the introductory courses in building a foundation for future student success.

4.8. Strategy 8. The curriculum should incorporate hands-on activities, teamwork, collaboration, and cooperation

We have combined teamwork and pair programming into our structured laboratory assignments. After evaluating the impact of cooperative learning on the laboratory assignments for three semesters, we now require all students to work in groups in the assignments. Each of our laboratory assignments also involves hands-on exploration of programming and problem solving.

4.9. Strategy 9. The curriculum should incorporate instructional and educational research

From the conception to the implementation of our integrated framework we have paid particular attention to research. All of our questions (in the placement examination and laboratory assignment post-tests) were rated in Bloom's taxonomy, so that we could determine what types of questions the students are best at solving. Our research designs are based on pre-post (repeated measures) comparisons or comparisons between control and treatment groups. We have also utilized sound statistical methods to validate the reliability and predictive validity of our results. Learning was measured by the placement examination, course examinations, and scores on laboratory assignment worksheets and post-tests. We also conducted qualitative interviews as part of a mixed method approach to obtain more in-depth results concerning the courseware, learning, and instruction. As a result of

incorporating instructional and educational research we have (1) revised and instituted a more valid and reliable placement examination, (2) revised the structure of our laboratory assignments, including now incorporating cooperative learning in all our CS1 laboratory assignments, (3) identified areas that need more emphasis in regular lectures (since post-test examination scores can be broken down by content area and Bloom's competence levels), and (4) documented that students enjoyed and benefited from the learning objects and are now working on delivering them to high school students for recruitment.

4.10. Strategy 10. The curriculum and its components should be institutionalized, particularly the monitoring and refinement processes to ensure continuity and quality

At present we have a project web site to store all documents and courseware. We have written technical papers to document our findings. Moreover, the Computer Science Education Department, the National Center for Information Technology in Education, and the College of Education and Human Sciences are committed to continue to fund and seek further funding to support this effort. The project team has direct access to the instructors and teaching assistants of the CS0, CS1, and CS2 courses. The entire process is documented and open and the department and faculty are informed of progress and status. To encourage faculty buy-in we have an instructional script for each laboratory assignment, and have put pre-tests and post-tests for each laboratory assignment online, for example. Delivering our tests and placement examination online also allow us to deliver these assessment mechanisms consistently over time, and make them convenient for an instructor to grade and tally scores. Online access is also convenient for the students. For example, high school seniors enrolling on CS1 can now take our placement examination remotely in the summer before attending school in the autumn semester. Further, we have had three different laboratory assignment instructors and two different course instructors for CS1 and three different laboratory assignment instructors and two different instructors for CS2. However, due to the institutionalization of the our placement examination and laboratory assignments, we have observed the following: (1) new instructors can ease into teaching CS1 and CS2 without too much of a "workload shock"; (2) faculty have been more willing to teach these courses; (3) instructors can now concentrate on their regular lectures instead of getting involved in the design and logistics of the laboratory assignments; (4) the department and students feel more confident in the consistency of the quality of the courses over different semesters and taught by different instructors.

Summary and Future Directions

We have presented a set of design strategies to address the needs of adapting to rapid and significant changes in the areas of IT and computing, of creating flexible, customizable, adaptable courseware, and of embedding educational research design

into the CS curriculum. Based on these strategies we have implemented an integrated framework that consists of a placement examination, three suites of structured laboratory assignments, learning objects, and evaluation/research designs. For each component we have described its design and implementation, as well as the results.

Our research showed positive achievement gains, across levels of Bloom's taxonomy and within specific content areas, for our reinvented CS1 course. The use of cooperative learning techniques within a CS laboratory environment increased student achievement. The learning objects were also shown to be an effective strategy to deliver CS content and to provide students with individualized practice opportunities.

The project has developed a process by which research experiments can be designed and conducted, students can be placed on different introductory CS courses, laboratory assignments can be monitored, revised, and replaced, course content can be flexibly delivered online, and valuable data can be collected via surveys and objective assessment mechanisms. With online materials and the support of the department the project will be able to impose tractability and continuity over time, allowing the curriculum to be further refined and evaluated and the students to be continuously assessed.

Our ongoing and future work includes the following. We are presently in the process of completing our design of the CS0 structured laboratory assignments. By the end of autumn 2005 we will have a complete set. This will complete all three suites of laboratory assignment materials. Our success with the online learning objects has led to plans to develop a suite of such objects, as well as online authoring tools to turn course content into SCORM-compliant learning objects. We are also in the process of using the placement examination to place students on CS0, CS1, or CS2. Once that is installed, we will conduct tests similar to those for placing students on CS0 or CS1, with the intent of also determining the validity and reliability of the examination in placing students on CS2. We also plan to conduct further experiments on student self-efficacy and motivation, to distinguish between majors and non-majors, to better determine whether majors are motivated and have more confidence in their knowledge of CS after each course. Our placement examination and structured laboratory assignments have been reviewed and revised during each semester. We have also outlined plans to deploy the two learning objects in high schools to recruit students into CS, to evaluate the learning objects, and to assess student knowledge.

In summary, the work conducted under the Reinventing Computer Science Project has resulted in three significant outcomes. First, the revised curriculum is effective; informal assessment by CS instructors indicates that students who take CS1 show a stronger background in problem solving using programming concepts on subsequent courses. Students who have taken these courses have also indicated that they have a greater confidence in their problem solving and programming abilities. Second, the project has produced substantial courseware and software tools that could be shared within the CS education community. (Readers are referred to our project web site,

http://cse.unl.edu/reinventCS, for downloads of the documents and materials described in this paper.) Third, the project has also put in place research components, including specific measures and educational research methods, to evaluate and validate new approaches used within the CS curriculum.

Acknowledgement

This work was supported in part by funding from the National Center for Information Technology in Education. We would like to thank Art Zygielbaum, Suzette Person, Rich Sincovec, Chuck Riedesel, Joseph Bernadt, Chao Chen, Kye Halsted, Brandon Hauff, Smitha Kasinadhuni, Andy Kosenander, Jeff Lang, Xuli Liu, Joyita Mallik, Saket Das, and Traci Fink for their invaluable help in implementing this project.

References

Advanced Distributed Learning (ADL). (2003, December 15). *DoD affirms SCORM'S role in training transformation* [Press release]. Retrieved May 12, 2004, from http://www.adlnet.org

Advanced Distributed Learning (ADL). (2004). *Sharable content object reference model (SCORM): 2004 overview.* Retrieved May 12, 2004, from http://www.adlnet.org/

Allan, V. H., & Kolesar, M. V. (1996). Teaching computer science: A problem solving approach that works. *Proceedings of the Annual National Educational Computing Conference* (ERIC Document Reproduction Service No. ED393878).

Anderson, R., & Roxa, T. (2000). Encouraging students in large classes. *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (SIGCSE'2000)*, 176 – 179. New York: ACM Press.

Bloom, B. S. (1956). *Taxonomy of educational objectives*, Book 1, *Cognitive domain*, New York: Longman.

Boster, F. J., Meyer, G. S., Roberto, A. J., & Inge, C. C. (2002). *A report on the effect of the united streaming application on educational performance.* Farmville, VA: Longwood University. Retrieved November 8, 2004 from http://caret.iste.org/index.cfm?StudyID=852&fuseaction=studySummary

Bradley, C., & Boyle, T. (2004). The design, development, and use of multimedia learning objects. *Journal of Educational Multimedia and Hypermedia*, *13*(4), 371 – 389.

Bruce, K. B. (2004). Controversy on how to teach CS 1: A discussion on the SIGCSE-members mailing list. *The SIGCSE Bulletin*, *36*(4), 29 – 34.

Cox, K., & Clark, D. (1998). The use of formative quizzes for deep learning. *Computers and Education*, *30*(3/4), 157 – 167.

Cross, J. H., Hendrix, T. D., & Barowski, L. A. (2002). Using the debugger as an integral part of teaching CS1. In D. Budny & G. Bjedov (Eds.), *Proceedings of the 32nd ASEE/IEEE Frontiers in Education Conference*, F1G-1 – FIG-6. Piscataway, NJ: IEEE Press.

Dabbagh, N. (1996). Creating personal relevance through adapting an educational task, situationally to a learners individual needs. *Proceedings of the National Convention of the Association for Educational Communications and Technology* (ERIC Document Reproduction Service No. ED397787).

Doran, M. V., & Langan, D. D. (1995). A cognitive-based approach to introductory computer science courses: lessons learned. *Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'95)*, 218 – 222. New York: ACM Press.

Gatfield, T. (1999). Examining student satisfaction with group projects and peer assessment. *Assessment and Evolution in Higher Education*, *24*(4), 365–377.

Geitz, R. (1994). Concepts in the classroom, programming in the lab. *Proceedings of the 25th SIGSE Symposium on Computer Science Education (SIGCSE'94)*, 164–168. New York: ACM Press.

Guzdial, M., & Soloway, E. (2002). Teaching the Nintendo generation to program. *Communications of the ACM*, *45*(4), 17–21.

Herrmann, N., Poppyack, J, Char, B, Zoski, P., Cera, C, Lass, R., et al. (2003). Redesigning introductory computer programming using multilevel online modules for a mixed audience. In S. Grissom, D. Knox, D. Joyce, & W. Dann (Eds.), *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (pp. 196–200). New York: ACM Press.

Herrmann, N., Poppyack, J, Char, B., & Zoski, P. (2004). Assessment of a course redesign: Introductory computer programming using online modules. In D. Joyce, D. Knox, W. Dann, & T. L. Naps (Eds.), *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 66–70. New York: ACM Press.

Jensen, M., Johnson D. W., & Johnson, R. T. (2002). Impact of positive interdependence during electronic quizzes on discourse and achievement. *Journal of Educational Research*, *95*(3), 161–166.

Johnson, D. W., & Johnson, R. T. (1989). *Cooperation and competition: Theory and research.* Edina, MS: Interaction Books.

Koppi, T., & Lavitt, N. (2003). Institutional use of learning objects three years on lessons. In E. Duval, W. Hodgins, D. Rehak, & R. Robson (Eds.), *ED-MEDIA 2003, Proceedings of Learning Objects Symposium: Lessons learned questions asked* (pp. 33–43). Norfolk, VA: Association for the Advancement of Computing in Education. Retrieved November 8, 2004, from http://www.aace.org/conf/edmedia/LO2003Symposium.pdf

Kumar, A. N. (2003). The effects of closed laboratory assignments in computer science I: an assessment. *Journal of Computing Sciences in Colleges*, *18*(5), 40–48.

Lischner, R. (2001). Explorations: structured laboratory assignments for first-time programmers. *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE'2001)*, 154–158. New York: ACM Press.

Longmire, W. (2000). A primer on learning objects. *Learning Circuits: ASTD's Online Magazine about E-learning.* Retrieved November 25, 2003, from http://learningcircuits.org

Lang, J., Nugent, G., Samal, A., & Soh, L.-K. (2006). Implementing CS1 with embedded instructional research design in laboratories. *IEEE Transactions on Education*, *49*(1), 157–165.

Malinger, M. (1998). Collaborative learning across borders: Dealing with student resistance. *Journal of Excellence in College Teaching*, *9*(1), 53–68.

Mayer, R. (2001). *Multimedia learning.* New YorkCambridge University Press.

McDowell, C., Werner, L., Bullock, E., & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. In A. Jacobs & F. Titsworth (Eds.), *Proceedings of the 25th International Conference on Software Engineering*, 602–607. Los Alamitos, CA: IEEE Computer Society.

Nugent, G. C. (2005). The use of learning objects in K–12: a public television perspective. *Tech Trends*, *49*(4), 61–66.

Nugent, G., Soh, L.-K., Samal, A., & Lang, J. (2006). A placement test for computer science: design, implementation, and analysis. *Computer Science Education*, *16*(1), 19–36.

Oliver, S. R., & Dalbey, J. (1994). A software development process laboratory for CS1 and CS2. *Proceedings of the 25th SIGSE Symposium on Computer Science Education (SIGCSE'94)*, 169–173. New York: ACM Press.

Parker, B. C., & McGregor, J. D. (1995). A goal-oriented approach to laboratory development and implementation. *Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'95)*, 92–96. New York: ACM Press.

Pasnik, S., & Nudell, H. (2003). *PBS K-12 digital classroom pilot evaluation report.* New York: Center for Children and Technology.

Pintrich, P. R., & DeGroot, E. V. (1990). Motivation and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology, 83*(1), 33–40.

Powers, K. (1999). A self-fulfilling prophecy: Online distance learning for introductory computing. *Proceedings of the National Educational Computing Conference 1999*. Washington, DC: ISTE. (ERIC Document Reproduction Service No. ED432994).

Prey, J. C. (1995). Cooperative learning in an undergraduate computer science curriculum [Electronic version]. In G. Bjedov & J. B. Perry (Eds.), *Proceedings of ASEE/IEEE 1995 Frontiers in Education Conference* (Session 3c23) (vol. 2, pp. 3c2.11–3c2.14). Retrieved November 8, 2004, from http://fie.engrng.pitt.edu/fie95/3c2/3c23/3c23.htm

Pugliese, L. C. (2002). The transformation of educational publishing: Emergence and growth of a teacher-centered, learning-object environment. *Technos*, *11*(3), 22–26.

Qin, Z., Johnson, D. W., & Johnson, R. T. (1995). Cooperative versus competitive efforts and problem solving. *Review of Educational Research*, *65*, 129–143.

Rebelsky, S. (2000). A web of resources for introductory computer science [Technical report]. (ERIC Document Reproduction Service No. ED445881).

Samal, A., Nugent, G., Soh, L.-K., Lang, J., & Person, S. (2005). Reinventing computer science curriculum at University of Nebraska. In L. M. PytlikZillig, M. Bodvarsson, & R. Bruning (Eds.), *Technology-based education: Bringing researchers and practitioners together* (ch. 4, pp. 63–82). Greenwich, CT: Information Age Publishing.

Schunk, D. H. (1981). Modeling and attributional effects on children's achievement: a self-efficacy analysis. *Journal of Educational Psychology*, *73*, 93–105.

Schunk, D. H. (1989). Self efficacy and cognitive skills learning. In R. Ames & C. Ames (Eds.), *Research on motivation in education: Goals and cognitions* (pp. 13–43). San Diego, CA: Academic Press.

Schunk, D. H., & Hanson, A. R. (1985). Peer models: Influence on children's self-efficacy and achievement. *Journal of Educational Psychology*, *77*, 313–322.

Sturm, D., & Moroh, M. (1994). Encouraging enrollment and retention of women in computer science classes. *Proceedings of the Annual National Educational Computing Conference*. Washington, DC: ISTE. (ERIC Document Reproduction Service No. ED396688).

Soh, L.-K., Samal, A., Person, S., Nugent, G., & Lang, J. (2005a). Closed laboratories with embedded instructional research design for CS1. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'2005)*, St. Louis, MO, February 23–27, 297–301.

Soh, L.-K., Samal, A., Person, S., Nugent, G., & Lang, J. (2005b). Analyzing relationships between closed laboratory assignments and course activities in CS1. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'2005)*, 183–187. New York: ACM Press.

Thweatt, M. (1994). CS1 closed laboratory assignment vs. open laboratory assignment experiment. *Proceedings of the 25th SIGSE Symposium on Computer Science Education*, 80–82. New York: ACM Press.

Urban-Lurain, M., & Weinshank, D. J. (1999, April). *Mastering computing technology: A new approach for non-computer science majors*. Paper presented at the Annual Meeting of the American Educational Research Association, Montreal, Quebec. (ERIC Document Reproduction Service No. ED347917).

Wiley, D. (2000). *Learning objects: Difficulties and opportunities* [Online publication], Retrieved May 12, 2004, from http://wiley.ed.usu.edu/articles.html

Williams, L. A., & Kessler, R. R. (2002). *Pair programming illuminated.* Boston: MA: Addison-Wesley.

Weber-Wullf, D. (2000). Combating the code warrior: A different sort of programming instruction. In D. Joyce (Ed.), *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education* (pp. 85–88). New York, NY: ACM Press.

Xenos, M., Pierrakeas, C., & Pintelas, P. (2002). A survey of student dropout rates and dropout causes concerning the students in the Course of Informatics of the Hellenic Open University. *Computers and Education*, *39*(4), 361–377.